Llama2-story-110m

준비물

- executorch
- 32GB 이상의 램
- docker 환경

과정

도커 환경 구성

1. 기본 환경: ubuntu 22.04

```
apt update
DEBIAN_FRONTEND=noninteractive TZ=Asia/Seoul apt install -y tzdata
apt install -y git build-essential cmake libideep-dev python3 python3-pip vim sudo python3-venv wget unzip zsh tmux git-lfs
```

2. executorch 설치

```
cd / && git clone https://github.com/pytorch/executorch.git
cd /executorch && git submodule update --init --recursive
cd /llama && python3 -m venv llama-env
source /llama/llama-env/bin/activate \
    && source /executorch/install_requirements.sh --pybind xnnpack
```

3. android 환경 셋업

```
apt install -y unzip zip curl libc++-dev
apt install -y zip openjdk-17-jdk

cd /root
wget https://dl.google.com/android/repository/commandlinetools-linux-11076708_latest.zip
unzip ./commandlinetools-linux-11076708_latest.zip

cd /root/cmdline-tools

./bin/sdkmanager --list --sdk_root=/opt/android-sdk
yes | ./bin/sdkmanager "platform-tools" "ndk;26.3.11579264" --sdk_root=/opt/android-sdk
yes | ./bin/sdkmanager "platforms;android-34"  --sdk_root=/opt/android-sdk
yes | ./bin/sdkmanager --licenses --sdk_root=/opt/android-sdk
```

Llama2-story 110m 모델 및 토크나이저 변환

1. 모델 및 토크나이저 다운로드

```
    cd ${WORKSPACE_DIRECTORY}/Llama2-110m
    wget
"https://huggingface.co/karpathy/tinyllamas/resolve/main/stories110M.pt"
    wget
"https://raw.githubusercontent.com/karpathy/llama2.c/master/tokenizer.model"
```

2. 모델 및 토크나이저 변환
   - parmas.json file

```
{
    "dim": 768,
    "multiple_of": 32,
    "n_heads": 12,
    "n_layers": 12,
    "norm_eps": 1e-05,
    "vocab_size": 32000
}
```

   - 변환 스크립트

```
    cd /executorch
    python -m examples.models.llama.export_llama -c
${WORKSPACE_DIRECTORY}/Llama2-110m/stories110M.pt -p
${WORKSPACE_DIRECTORY}/Llama2-110m/params.json -X -kv -o
${WORKSPACE_DIRECTORY}/Llama2-110m/llama2-story.pte

    python -m extension.llm.tokenizer.tokenizer -t
${WORKSPACE_DIRECTORY}/Llama2-110m/tokenizer.model -o
${WORKSPACE_DIRECTORY}/Llama2-110m/llama2-story-tokenizer.bin
```

변환된 모델 실행 환경 구성

1. x86-pc 용 실행기 컴파일

```
cmake -DPYTHON_EXECUTABLE=python \
    -DCMAKE_INSTALL_PREFIX=cmake-out \
    -DEXECUTORCH_ENABLE_LOGGING=1 \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_EXTENSION_DATA_LOADER=ON \
    -DEXECUTORCH_BUILD_EXTENSION_MODULE=ON \
    -DEXECUTORCH_BUILD_EXTENSION_TENSOR=ON \
    -DEXECUTORCH_BUILD_XNNPACK=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out .

cmake --build cmake-out -j16 --target install --config Release
cmake -DPYTHON_EXECUTABLE=python \
    -DCMAKE_INSTALL_PREFIX=cmake-out \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_XNNPACK=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -Bcmake-out/examples/models/llama \
    examples/models/llama

cmake --build cmake-out/examples/models/llama -j16 --config Release
```

2. android device용 실행기 컴파일

```
export ANDROID_NDK=/opt/android-sdk/ndk/26.3.11579264

cmake -DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake
\
    -DANDROID_ABI=arm64-v8a \
    -DANDROID_PLATFORM=android-23 \
    -DCMAKE_INSTALL_PREFIX=cmake-out-android \
    -DCMAKE_BUILD_TYPE=Release \
    -DEXECUTORCH_BUILD_EXTENSION_DATA_LOADER=ON \
    -DEXECUTORCH_BUILD_EXTENSION_MODULE=ON \
    -DEXECUTORCH_BUILD_EXTENSION_TENSOR=ON \
    -DEXECUTORCH_ENABLE_LOGGING=1 \
    -DPYTHON_EXECUTABLE=python \
    -DEXECUTORCH_BUILD_XNNPACK=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out-android .

cmake --build cmake-out-android -j16 --target install --config Release

cmake  -
DCMAKE_TOOLCHAIN_FILE=$ANDROID_NDK/build/cmake/android.toolchain.cmake \
    -DANDROID_ABI=arm64-v8a \
    -DANDROID_PLATFORM=android-23 \
    -DCMAKE_INSTALL_PREFIX=cmake-out-android \
    -DCMAKE_BUILD_TYPE=Release \
    -DPYTHON_EXECUTABLE=python \
    -DEXECUTORCH_BUILD_XNNPACK=ON \
    -DEXECUTORCH_BUILD_KERNELS_OPTIMIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_QUANTIZED=ON \
    -DEXECUTORCH_BUILD_KERNELS_CUSTOM=ON \
    -Bcmake-out-android/examples/models/llama \
    examples/models/llama

cmake --build cmake-out-android/examples/models/llama -j16 --config Release
```

변환된 모델 실행

PC의 경우

- 다음 명령어 실행

```
    cd /executorch/cmake-out/examples/models/llama
    ./llama_main --model_path   ${WORKSPACE_DIRECTORY}/Llama2-110m/xnnpack_dq_llama2.pte \
               --tokenizer_path ${WORKSPACE_DIRECTORY}/Llama2-110m/llama2-story-tokenizer.bin
```

안드로이드의 경우

1. 모델 업로드

```
    adb shell mkdir /data/local/tmp/llama2-story

    adb push ${WORKSPACE_DIRECTORY}/Llama2-110m/llama2-story-tokenizer.bin /data/local/tmp/llama2-story/llama2-story-tokenizer.bin

    adb push llama2-story.pte /data/local/tmp/llama2-story/
```

2. 실행기 업로드

```
    cd /executorch/cmake-out-android/examples/models/llama
    adb push llama_main /data/local/tmp/llama2-story/
    adb shell
    # 여기서부터 안드로이드 쉘
    cd /data/local/tmp/llama2-story
    ./llama_main --model_path llama2-story.pte  --tokenizer_path ./llama2-story-tokenizer.bin --prompt "[user]: Explain who you are\n[bot]:"
```